



US008054765B2

(12) **United States Patent**
Passey et al.

(10) **Patent No.:** **US 8,054,765 B2**
(45) **Date of Patent:** **Nov. 8, 2011**

(54) **SYSTEMS AND METHODS FOR PROVIDING VARIABLE PROTECTION**

(75) Inventors: **Aaron J. Passey**, Seattle, WA (US); **Neal T. Fachan**, Seattle, WA (US)

(73) Assignee: **EMC Corporation**, Hopkinton, MA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 29 days.

(21) Appl. No.: **12/484,905**

(22) Filed: **Jun. 15, 2009**

(65) **Prior Publication Data**

US 2009/0252066 A1 Oct. 8, 2009

Related U.S. Application Data

(63) Continuation of application No. 11/256,410, filed on Oct. 21, 2005, now Pat. No. 7,551,572.

(51) **Int. Cl.**
H04L 12/28 (2006.01)

(52) **U.S. Cl.** **370/256; 370/382**

(58) **Field of Classification Search** **370/217, 370/256, 379, 382, 408; 707/7, 100, 101, 707/103; 711/162; 714/36, 718**

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,163,131 A	11/1992	Row et al.
5,181,162 A	1/1993	Smith et al.
5,212,784 A	5/1993	Sparks
5,230,047 A	7/1993	Frey et al.
5,251,206 A	10/1993	Calvignac et al.

5,258,984 A	11/1993	Menon et al.
5,329,626 A	7/1994	Klein et al.
5,359,594 A	10/1994	Gould et al.
5,403,639 A	4/1995	Belsan et al.
5,459,871 A	10/1995	Van Den Berg
5,481,699 A	1/1996	Saether
5,548,724 A	8/1996	Akizawa et al.
5,548,795 A	8/1996	Au
5,568,629 A	10/1996	Gentry et al.
5,596,709 A	1/1997	Bond et al.
5,606,669 A	2/1997	Bertin et al.
5,612,865 A	3/1997	Dasgupta
5,649,200 A	7/1997	Leblang et al.
5,657,439 A	8/1997	Jones et al.
5,668,943 A	9/1997	Attanasio et al.
5,680,621 A	10/1997	Korenshtein
5,694,593 A	12/1997	Baclawski
5,696,895 A	12/1997	Hemphill et al.
5,734,826 A	3/1998	Olnowich et al.
5,754,756 A	5/1998	Watanabe et al.

(Continued)

FOREIGN PATENT DOCUMENTS

EP 0774723 5/1997

(Continued)

OTHER PUBLICATIONS

Nov. 15, 2002 Int'l Search report PCT/US02/24728, 2 pages.

(Continued)

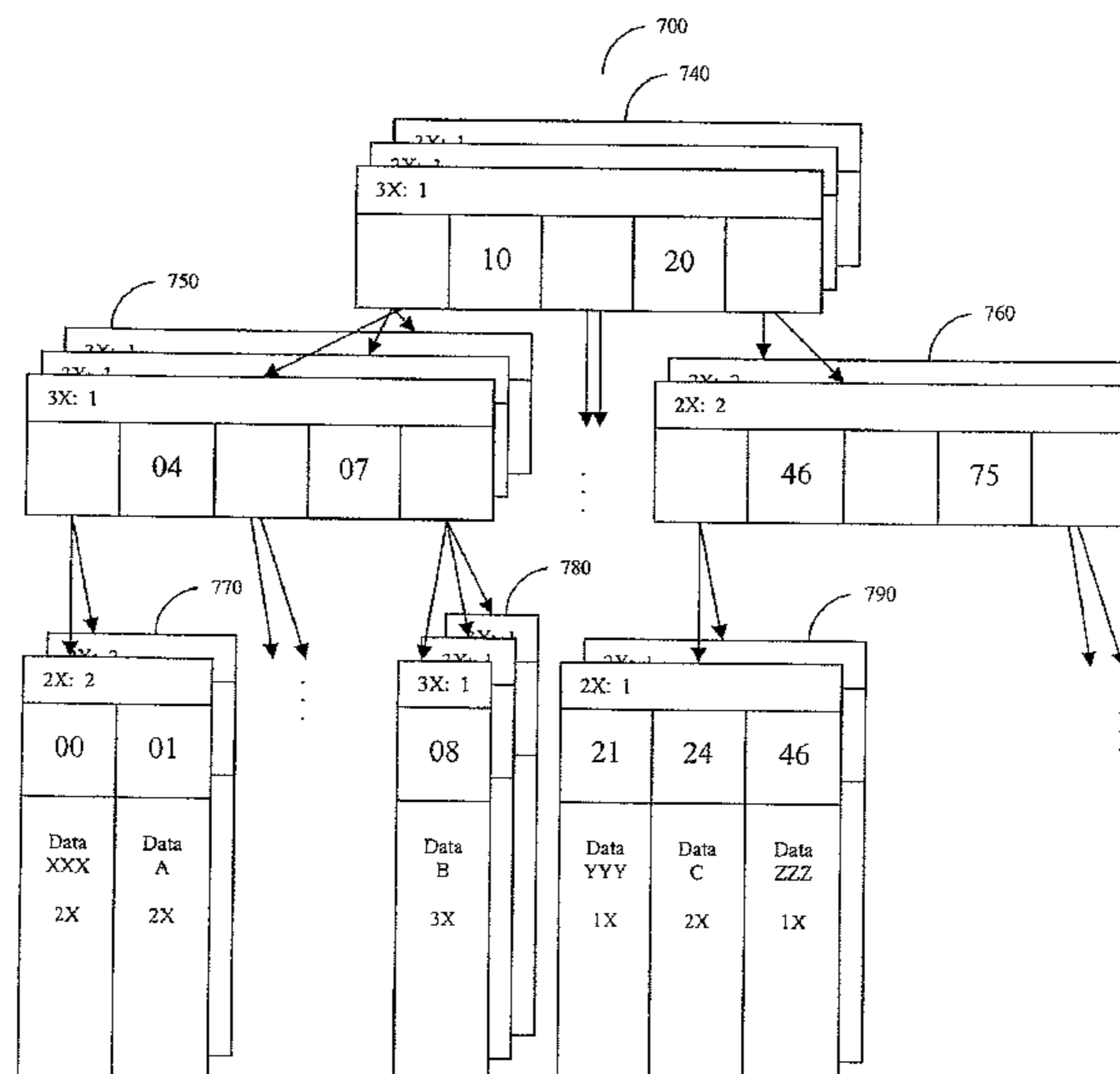
Primary Examiner — Andrew Lai

(74) *Attorney, Agent, or Firm* — Knobbe Martens Olson & Bear LLP

(57) **ABSTRACT**

The systems and methods maintain varying protection levels of related objects, such as when nodes of a data structure are protected at the same or greater protection level as their children nodes. In one embodiment, the nodes store data to track protection levels of their children nodes.

14 Claims, 8 Drawing Sheets



U.S. PATENT DOCUMENTS							
5,761,659	A	6/1998	Bertoni	6,618,798	B1	9/2003	Burton et al.
5,774,643	A	6/1998	Lubbers et al.	6,631,411	B1	10/2003	Welter et al.
5,799,305	A	8/1998	Bortvedt et al.	6,658,554	B1	12/2003	Moshovos et al.
5,805,578	A	9/1998	Stirpe et al.	6,662,184	B1	12/2003	Friedberg
5,805,900	A	9/1998	Fagen et al.	6,671,686	B2	12/2003	Pardon et al.
5,806,065	A	9/1998	Lomet	6,671,704	B1	12/2003	Gondi et al.
5,822,790	A	10/1998	Mehrotra	6,671,772	B1	12/2003	Cousins
5,862,312	A	1/1999	Mann	6,687,805	B1	2/2004	Cochran
5,870,563	A	2/1999	Roper et al.	6,725,392	B1	4/2004	Frey et al.
5,878,410	A	3/1999	Zbikowski et al.	6,732,125	B1	5/2004	Autrey et al.
5,878,414	A	3/1999	Hsiao et al.	6,742,020	B1	5/2004	Dimitroff et al.
5,884,046	A	3/1999	Antonov	6,748,429	B1	6/2004	Talluri et al.
5,884,098	A	3/1999	Mason, Jr.	6,801,949	B1	10/2004	Bruck et al.
5,884,303	A	3/1999	Brown	6,848,029	B2	1/2005	Coldewey
5,890,147	A	3/1999	Peltonen et al.	6,856,591	B1	2/2005	Ma et al.
5,917,998	A	6/1999	Cabrera et al.	6,871,295	B2	3/2005	Ulrich et al.
5,933,834	A	8/1999	Aichelen	6,895,482	B1	5/2005	Blackmon et al.
5,943,690	A	8/1999	Dorricott et al.	6,895,534	B2	5/2005	Wong et al.
5,963,963	A	10/1999	Schmuck et al.	6,907,011	B1	6/2005	Miller et al.
5,966,707	A	10/1999	Van Huben et al.	6,907,520	B2	6/2005	Parady
5,996,089	A	11/1999	Mann	6,917,942	B1	7/2005	Burns et al.
6,000,007	A	12/1999	Leung et al.	6,920,494	B2	7/2005	Heitman et al.
6,014,669	A	1/2000	Slaughter et al.	6,922,696	B1 *	7/2005	Lincoln et al. 707/101
6,021,414	A	2/2000	Fuller	6,922,708	B1	7/2005	Sedlar
6,029,168	A	2/2000	Frey	6,934,878	B2	8/2005	Massa et al.
6,038,570	A	3/2000	Hitz et al.	6,940,966	B2	9/2005	Lee
6,044,367	A	3/2000	Wolff	6,954,435	B2	10/2005	Billhartz et al.
6,052,759	A	4/2000	Stallmo et al.	6,990,604	B2	1/2006	Binger
6,055,543	A	4/2000	Christensen et al.	6,990,611	B2	1/2006	Busser
6,055,564	A	4/2000	Phaal	7,007,044	B1	2/2006	Rafert et al.
6,070,172	A	5/2000	Lowe	7,007,097	B1	2/2006	Huffman et al.
6,081,833	A	6/2000	Okamoto et al.	7,017,003	B2	3/2006	Murotani et al.
6,081,883	A	6/2000	Popelka et al.	7,043,485	B2	5/2006	Manley et al.
6,108,759	A	8/2000	Orcutt et al.	7,043,567	B2	5/2006	Trantham
6,117,181	A	9/2000	Dearth et al.	7,069,320	B1	6/2006	Chang et al.
6,122,754	A	9/2000	Litwin et al.	7,103,597	B2	9/2006	McGoveran
6,138,126	A	10/2000	Hitz et al.	7,111,305	B2	9/2006	Solter et al.
6,154,854	A	11/2000	Stallmo	7,113,938	B2	9/2006	Highleyman et al.
6,173,374	B1	1/2001	Heil et al.	7,124,264	B2	10/2006	Yamashita
6,202,085	B1	3/2001	Benson et al.	7,146,524	B2	12/2006	Patel et al.
6,209,059	B1	3/2001	Ofer et al.	7,152,182	B2	12/2006	Ji et al.
6,219,693	B1	4/2001	Napolitano et al.	7,177,295	B1	2/2007	Sholander et al.
6,226,377	B1	5/2001	Donaghue, Jr.	7,181,746	B2	2/2007	Perycz et al.
6,279,007	B1	8/2001	Uppala	7,184,421	B1	2/2007	Liu et al.
6,321,345	B1	11/2001	Mann	7,194,487	B1	3/2007	Kekre et al.
6,334,168	B1	12/2001	Islam et al.	7,206,805	B1	4/2007	McLaughlin, Jr.
6,353,823	B1	3/2002	Kumar	7,225,204	B2	5/2007	Manley et al.
6,384,626	B2	5/2002	Tsai et al.	7,228,299	B1	6/2007	Harmer et al.
6,385,626	B1	5/2002	Tamer et al.	7,240,235	B2	7/2007	Lewalski-Brechter
6,393,483	B1	5/2002	Latif et al.	7,249,118	B2	7/2007	Sandler et al.
6,397,311	B1	5/2002	Capps	7,257,257	B2	8/2007	Anderson et al.
6,405,219	B2	6/2002	Saether et al.	7,290,056	B1	10/2007	McLaughlin, Jr.
6,408,313	B1	6/2002	Campbell et al.	7,313,614	B2	12/2007	Considine et al.
6,415,259	B1	7/2002	Wolfinger et al.	7,318,134	B1	1/2008	Oliveira et al.
6,421,781	B1	7/2002	Fox et al.	7,346,346	B2	3/2008	Lipsit
6,434,574	B1	8/2002	Day et al.	7,346,720	B2	3/2008	Fachan
6,449,730	B2	9/2002	Mann	7,370,064	B2	5/2008	Yousefi'zadeh
6,453,389	B1	9/2002	Weinberger et al.	7,373,426	B2	5/2008	Jinmei et al.
6,457,139	B1	9/2002	D'Errico et al.	7,386,675	B2	6/2008	Fachan
6,463,442	B1	10/2002	Bent et al.	7,386,697	B1	6/2008	Case et al.
6,496,842	B1	12/2002	Lyness	7,440,966	B2	10/2008	Adkins et al.
6,499,091	B1	12/2002	Bergsten	7,451,341	B2	11/2008	Okaki et al.
6,502,172	B2	12/2002	Chang	7,509,448	B2	3/2009	Fachan et al.
6,502,174	B1	12/2002	Beardsley	7,509,524	B2	3/2009	Patel et al.
6,523,130	B1	2/2003	Hickman et al.	7,533,298	B2	5/2009	Smith et al.
6,526,478	B1	2/2003	Kirby	7,546,354	B1	6/2009	Fan et al.
6,546,443	B1	4/2003	Kakivaya et al.	7,546,412	B2	6/2009	Ahmad et al.
6,549,513	B1	4/2003	Chao et al.	7,551,572	B2	6/2009	Passey et al.
6,557,114	B2	4/2003	Mann	7,558,910	B2	7/2009	Alverson et al.
6,567,894	B1	5/2003	Hsu et al.	7,571,348	B2	8/2009	Deguchi et al.
6,567,926	B2	5/2003	Mann	7,577,258	B2 *	8/2009	Wiseman et al. 380/281
6,571,244	B1 *	5/2003	Larson 707/753	7,577,667	B2	8/2009	Hinshaw et al.
6,571,349	B1	5/2003	Mann	7,590,652	B2	9/2009	Passey et al.
6,574,745	B2	6/2003	Mann	7,593,938	B2	9/2009	Lemar et al.
6,594,655	B2	7/2003	Tal et al.	7,596,713	B2	9/2009	Mani-Meitav et al.
6,594,660	B1	7/2003	Berkowitz et al.	7,631,066	B1	12/2009	Schatz et al.
6,594,744	B1	7/2003	Humlicek et al.	7,665,123	B1	2/2010	Szor et al.
6,598,174	B1	7/2003	Parks et al.	7,676,691	B2	3/2010	Fachan et al.
				7,680,836	B2	3/2010	Anderson et al.

US 8,054,765 B2

7,680,842 B2	3/2010	Anderson et al.	2004/0117802 A1	6/2004	Green	
7,685,126 B2	3/2010	Patel et al.	2004/0133670 A1	7/2004	Kaminsky et al.	
7,685,162 B2	3/2010	Heider et al.	2004/0143647 A1	7/2004	Cherkasova	
7,689,597 B1	3/2010	Bingham et al.	2004/0153479 A1	8/2004	Mikesell et al.	
7,707,193 B2	4/2010	Zayas et al.	2004/0158549 A1	8/2004	Matena et al.	
7,716,262 B2	5/2010	Pallapotu	2004/0174798 A1*	9/2004	Riguidel et al.	369/111
7,734,603 B1	6/2010	McManis	2004/0189682 A1	9/2004	Troyansky et al.	
7,739,288 B2	6/2010	Lemar et al.	2004/0199734 A1	10/2004	Rajamani et al.	
7,743,033 B2	6/2010	Patel et al.	2004/0199812 A1	10/2004	Earl et al.	
7,752,402 B2	7/2010	Fachan et al.	2004/0205141 A1	10/2004	Goland	
7,756,898 B2	7/2010	Passey et al.	2004/0230748 A1	11/2004	Ohba	
7,779,048 B2	8/2010	Fachan et al.	2004/0240444 A1	12/2004	Matthews et al.	
2001/0042224 A1	11/2001	Stanfill et al.	2004/0260673 A1	12/2004	Hitz et al.	
2001/0047451 A1	11/2001	Noble et al.	2004/0267747 A1	12/2004	Choi et al.	
2001/0056492 A1	12/2001	Bressoud et al.	2005/0010592 A1	1/2005	Guthrie	
2002/0010696 A1	1/2002	Izumi	2005/0033778 A1	2/2005	Price	
2002/0029200 A1	3/2002	Dulin et al.	2005/0044197 A1	2/2005	Lai	
2002/0035668 A1	3/2002	Nakano et al.	2005/0066095 A1	3/2005	Mullick et al.	
2002/0038436 A1	3/2002	Suzuki	2005/0114402 A1	5/2005	Guthrie	
2002/0049778 A1	4/2002	Bell et al.	2005/0114609 A1	5/2005	Shorb	
2002/0055940 A1	5/2002	Elkan	2005/0125456 A1	6/2005	Hara et al.	
2002/0072974 A1	6/2002	Pugliese et al.	2005/0131860 A1	6/2005	Livshits	
2002/0075870 A1	6/2002	de Azevedo et al.	2005/0131990 A1	6/2005	Jewell	
2002/0078161 A1	6/2002	Cheng	2005/0138195 A1	6/2005	Bono	
2002/0078180 A1	6/2002	Miyazawa	2005/0138252 A1	6/2005	Gwilt	
2002/0083078 A1	6/2002	Pardon et al.	2005/0171960 A1	8/2005	Lomet	
2002/0083118 A1	6/2002	Sim	2005/0171962 A1	8/2005	Martin et al.	
2002/0087366 A1	7/2002	Collier et al.	2005/0187889 A1	8/2005	Yasoshima	
2002/0095438 A1	7/2002	Rising et al.	2005/0188052 A1	8/2005	Ewanchuk et al.	
2002/0107877 A1	8/2002	Whiting et al.	2005/0192993 A1	9/2005	Messinger	
2002/0124137 A1	9/2002	Ulrich et al.	2005/0289169 A1	12/2005	Adya et al.	
2002/0138559 A1	9/2002	Ulrich et al.	2005/0289188 A1	12/2005	Nettleton et al.	
2002/0156840 A1	10/2002	Ulrich et al.	2006/0004760 A1	1/2006	Clift et al.	
2002/0156891 A1	10/2002	Ulrich et al.	2006/0041894 A1	2/2006	Cheng	
2002/0156973 A1	10/2002	Ulrich et al.	2006/0047713 A1	3/2006	Gornshtein et al.	
2002/0156974 A1	10/2002	Ulrich et al.	2006/0047925 A1	3/2006	Perry	
2002/0156975 A1	10/2002	Staub et al.	2006/0053263 A1*	3/2006	Prahlad et al.	711/162
2002/0158900 A1	10/2002	Hsieh et al.	2006/0059467 A1	3/2006	Wong	
2002/0161846 A1	10/2002	Ulrich et al.	2006/0074922 A1	4/2006	Nishimura	
2002/0161850 A1	10/2002	Ulrich et al.	2006/0083177 A1	4/2006	Iyer et al.	
2002/0161973 A1	10/2002	Ulrich et al.	2006/0095438 A1	5/2006	Fachan et al.	
2002/0163889 A1	11/2002	Yemini et al.	2006/0101062 A1	5/2006	Godman et al.	
2002/0165942 A1	11/2002	Ulrich et al.	2006/0129584 A1	6/2006	Hoang et al.	
2002/0166026 A1	11/2002	Ulrich et al.	2006/0129631 A1	6/2006	Na et al.	
2002/0166079 A1	11/2002	Ulrich et al.	2006/0129983 A1	6/2006	Feng	
2002/0169827 A1	11/2002	Ulrich et al.	2006/0155831 A1	7/2006	Chandrasekaran	
2002/0170036 A1	11/2002	Cobb et al.	2006/0206536 A1	9/2006	Sawdon et al.	
2002/0174295 A1	11/2002	Ulrich et al.	2006/0230411 A1	10/2006	Richter et al.	
2002/0174296 A1	11/2002	Ulrich et al.	2006/0277432 A1	12/2006	Patel	
2002/0178162 A1	11/2002	Ulrich et al.	2006/0288161 A1	12/2006	Cavallo	
2002/0191311 A1	12/2002	Ulrich et al.	2006/0294589 A1	12/2006	Achanta et al.	
2002/0194523 A1	12/2002	Ulrich et al.	2007/0038887 A1	2/2007	Witte et al.	
2002/0194526 A1	12/2002	Ulrich et al.	2007/0091790 A1	4/2007	Passey et al.	
2002/0198864 A1	12/2002	Ostermann et al.	2007/0094269 A1	4/2007	Mikesell et al.	
2003/0005159 A1	1/2003	Kumhyr	2007/0094277 A1	4/2007	Fachan et al.	
2003/0009511 A1	1/2003	Giotta et al.	2007/0094310 A1	4/2007	Passey et al.	
2003/0014391 A1	1/2003	Evans et al.	2007/0094431 A1	4/2007	Fachan	
2003/0033308 A1	2/2003	Patel et al.	2007/0094449 A1	4/2007	Allison et al.	
2003/0061491 A1	3/2003	Jaskiewicz et al.	2007/0094452 A1	4/2007	Fachan	
2003/0109253 A1	6/2003	Fenton et al.	2007/0124337 A1	5/2007	Flam	
2003/0120863 A1	6/2003	Lee et al.	2007/0168351 A1	7/2007	Fachan	
2003/0125852 A1	7/2003	Schade et al.	2007/0171919 A1	7/2007	Godman et al.	
2003/0126522 A1*	7/2003	English et al.	2007/0192254 A1	8/2007	Hinkle	714/718
2003/0131860 A1	7/2003	Ashcraft et al.	2007/0195810 A1	8/2007	Fachan	
2003/0135514 A1	7/2003	Patel et al.	2007/0233684 A1	10/2007	Verma et al.	
2003/0149750 A1	8/2003	Franzenburg	2007/0233710 A1	10/2007	Passey et al.	
2003/0158873 A1	8/2003	Sawdon et al.	2007/0244877 A1	10/2007	Kempka	
2003/0161302 A1	8/2003	Zimmermann et al.	2007/0255765 A1	11/2007	Robinson	
2003/0163726 A1*	8/2003	Kidd	2008/0005145 A1	1/2008	Worrall	713/200
2003/0172149 A1	9/2003	Edsall et al.	2008/0010507 A1	1/2008	Vingralek	
2003/0177308 A1	9/2003	Lewalski-Brechter	2008/0021907 A1	1/2008	Patel et al.	
2003/0182312 A1	9/2003	Chen et al.	2008/0031238 A1	2/2008	Harmelin et al.	
2003/0182325 A1	9/2003	Manley et al.	2008/0034004 A1	2/2008	Cisler et al.	
2003/0233385 A1	12/2003	Srinivasa et al.	2008/0044016 A1	2/2008	Henzinger	
2004/0003053 A1	1/2004	Williams	2008/0046432 A1	2/2008	Anderson et al.	
2004/0024731 A1	2/2004	Cabrera et al.	2008/0046443 A1	2/2008	Fachan et al.	
2004/0024963 A1	2/2004	Talagala et al.	2008/0046444 A1	2/2008	Fachan et al.	
2004/0078680 A1*	4/2004	Hu et al.	2008/0046445 A1	2/2008	Passey et al.	714/36
2004/0078812 A1	4/2004	Calvert	2008/0046475 A1	2/2008	Anderson et al.	

2008/0046476	A1	2/2008	Anderson et al.
2008/0046667	A1	2/2008	Fachan et al.
2008/0059541	A1	3/2008	Fachan et al.
2008/0059734	A1	3/2008	Mizuno
2008/0126365	A1	5/2008	Fachan et al.
2008/0151724	A1	6/2008	Anderson et al.
2008/0154978	A1	6/2008	Lemar et al.
2008/0155191	A1	6/2008	Anderson et al.
2008/0168304	A1	7/2008	Flynn et al.
2008/0168458	A1	7/2008	Fachan et al.
2008/0243773	A1	10/2008	Patel et al.
2008/0256103	A1	10/2008	Fachan et al.
2008/0256537	A1	10/2008	Fachan et al.
2008/0256545	A1	10/2008	Fachan et al.
2008/0294611	A1	11/2008	Anglin et al.
2009/0055399	A1	2/2009	Lu et al.
2009/0055604	A1	2/2009	Lemar et al.
2009/0055607	A1	2/2009	Schack et al.
2009/0125563	A1	5/2009	Wong et al.
2009/0210880	A1	8/2009	Fachan et al.
2009/0248756	A1	10/2009	Akidau et al.
2009/0248765	A1	10/2009	Akidau et al.
2009/0248975	A1	10/2009	Daud et al.
2009/0249013	A1	10/2009	Daud et al.
2009/0327218	A1	12/2009	Passey et al.
2010/0011011	A1	1/2010	Lemar et al.
2010/0122057	A1	5/2010	Strumpfen et al.
2010/0161556	A1	6/2010	Anderson et al.
2010/0161557	A1	6/2010	Anderson et al.
2010/0185592	A1	7/2010	Kryger
2010/0223235	A1	9/2010	Fachan
2010/0235413	A1	9/2010	Patel
2010/0241632	A1	9/2010	Lemar et al.
2010/0306786	A1	12/2010	Passey
2011/0016353	A1	1/2011	Mikesell et al.
2011/0022790	A1	1/2011	Fachan
2011/0035412	A1	2/2011	Fachan
2011/0044209	A1	2/2011	Fachan
2011/0060779	A1	3/2011	Lemar et al.

FOREIGN PATENT DOCUMENTS

EP	1421520	5/2004
EP	1563411	8/2005
EP	2284735	2/2011
EP	2299375	3/2011
JP	04096841	3/1992
JP	2006-506741	6/2004
JP	4464279	5/2010
JP	4504677	7/2010
WO	WO 94/29796	12/1994
WO	WO 00/57315	9/2000
WO	WO 01/14991	3/2001
WO	WO 01/33829	5/2001
WO	WO 02/061737	8/2002
WO	WO 03/012699	2/2003
WO	WO 2004/046971	6/2004
WO	WO 2008/021527	2/2008
WO	WO 2008/021528	2/2008
WO	WO 2008/127947	10/2008

OTHER PUBLICATIONS

Apr. 20, 2004 Int'l Search report PCT/US03/36699, 10 pages.
 Aug. 6, 2004 Int'l Search report PCT/US03/33704, 11 pages.
 May 21, 2007 European Search Report EP 02756944.1-2201, 8 pages.
 Feb. 22, 2008 Int'l Search report PCT/US07/018326, 20 pages.
 May 8, 2008 Int'l Search report PCT/US07/018324, 13 pages.
 Philip A. Bernstein, Vassos Hadzilacos, and Nathan Goodman, *Concurrency Control and Recovery in Database Systems*, Addison-Wesley, 370 pages, 1987.
 Kenneth P. Birman, *Building Secure and Reliable Network Applications*, Manning, 1996, pp. 1-327.
 Kenneth P. Birman, *Building Secure and Reliable Network Applications*, Manning, 1996, pp. 328-619.
 Birk, Y., *Deterministic load-balancing schemes for disk-based video-on-demand storage servers*, *Mass Storage Systems 1995; Storage—At the Forefront of Information Infrastructures*, Proceedings of the

Fourteenth IEEE Symposium on Monterey, CA, Sep. 11-14, 1995, pp. 17-25.
 Coulouris et al., *Distributed Systems Concepts and Design*; Addison-Wesley, Second Edition, 1994, pp. 353-371 and 377-405.
 Jim Gray and Andreas Reuter, *Transaction Processing: Concepts and Techniques*, Morgan Kaufmann, 1993, pp. 1-328.
 Jim Gray and Andreas Reuter, *Transaction Processing: Concepts and Techniques*, Morgan Kaufmann, 1993, pp. 329-664.
 Jim Gray and Andreas Reuter, *Transaction Processing: Concepts and Techniques*, Morgan Kaufmann, 1993, pp. 665-1105.
 Kumar, Akhil, "An analysis of borrowing policies for escrow transactions in a replicated data environment", *Institute of Electrical and Electronics Engineers; Proceedings of the International Conference on Data Engineering*, Los Angeles, Feb. 5-9, 1990; Los Alamitos, IEEE, Comp. Soc. Press, US, vol. Conf. 6, Feb. 5, 1990, pp. 446-454, XP010018185 ISBN: 978-0-8186-2025-6 p. 446, left-hand column, line 1-p. 447, last line.
 Nancy A. Lynch, *Distributed Algorithms*, Morgan Kaufmann, 1996, pp. 1-409.
 Nancy A. Lynch, *Distributed Algorithms*, Morgan Kaufmann, 1996 pp. 410-871.
 Sape Mullender, editor, *Distributed Systems (2nd Ed.)*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1993, pp. 1-327.
 Sape Mullender, editor, *Distributed Systems (2nd Ed.)*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1993, pp. 328-604.
 Gibson, Garth A. et al., *The Scotch Parallel Storage System*, IEEE, 1995, pp. 403-410.
 Gerhard Weikum, et al., *Transactional Information Systems: Theory, Algorithms, and the Practice of Concurrency Control and Recovery*, Morgan Kaufmann, 2002, pp. 1-450.
 Gerhard Weikum, et al., *Transactional Information Systems: Theory, Algorithms, and the Practice of Concurrency Control and Recovery*, Morgan Kaufmann, 2002, pp. 451-863.
 Sanjay Ghemawat et al., *The Google File System*, *Symposium on Operating Systems Principles*, Oct. 19-22, 2003, 15 pages, Bolton Landing, NY.
 Pei Cao et al., *The TickerTAIP parallel RAID architecture*, 1993, 12 pages, Princeton, NJ.
 Pei Cao et al., *The TickerTAIP parallel RAID architecture*, *ACM Transactions on Computer Systems*, vol. 12, No. 3, Aug. 1994, 34 pages, Palto Alto, California.
 Duzett, Bob et al., *An Overview of the nCube 3 Supercomputer*, IEEE, Jul. 1992, pp. 458-464.
 Hartman, John Henry, *The Zebra Striped Network File System*, (Sc.B. (Brown University) 1987, pp. 1-148.
 Long, Darrell D.E., et al., *Swift/RAID: A Distributed RAID System*, *Computing Systems*, vol. 7, No. 3 Summer 1994, pp. 333-359.
 Stallings, William, *Operating Systems*, Prentice Hall, Fourth Edition; 2001, pp. 197-253 and 265-293.
 Michael Stonebraker et al., *Distributed Raid: A New Multiple Copy Algorithm*, *Proc. Sixth Int. Conf. Data Eng.*, IEEE Service Center, Piscataway, NJ (IEEE cat No. 90CH2840-7), pp. 430-437, Feb. 5, 1990.
 United States Court of Appeals, Federal Circuit, *Seachange International, Inc. vs. nCUBE, Corp.*, Oct. 26, 2004, 28 pages.
 United States Court of Appeals, Federal Circuit, *Seachange International, Inc. vs. C-Corp, Inc.*, Jun. 29, 2005, 22 pages.
 United States District Court, Delaware, *Seachange International, Inc. vs. nCUBE, Corp.*, Apr. 7, 2004, 13 pages.
 United States District Court, Delaware, *Seachange International, Inc. vs. nCUBE, Corp.*, Aug. 29, 2000, 12 pages.
 Steven M. Bauer, Letter to Steve Goldman, Jul. 31, 2006 (including the first page of the patents), 8 pages.
 Arthur S. Rose, Letter to Steven M. Bauer, Aug. 7, 2006, 2 pages.
 Steven M. Bauer, Letter to Arthur S. Rose, Sep. 25, 2006, 2 pages.
 Steven M. Bauer, Letter to Arthur S. Rose, Oct. 31, 2006, 1 page.
 Arthur S. Rose, Letter to Steven M. Bauer, Nov. 9, 2006, 2 pages.
 Steven M. Bauer, Letter to Arthur S. Rose, Feb. 6, 2007, 2 pages.
 Arthur S. Rose, Letter to Steven M. Bauer, Feb. 23, 2007, 1 page.
 Steven M. Bauer, Letter to Arthur S. Rose, Jun. 7, 2007(including claim chart), 28 pages.

- Arthur S. Rose, Letter to Steven M. Bauer, Jun. 22, 2007, 1 page.
- Todd A. Gerety, Letter to Arthur S. Rose, Jun. 26, 2007 (exhibits listed separately below), 1 page.
- Isilon Systems, "Isilon IQ Platform Overview", 4 pages, 2007 (Exhibit A).
- Isilon Systems, "Uncompromising Reliability through Clustered Storage", Sep. 2006, 10 pages (Exhibit B).
- Byteandswitch, "Discovery Chooses Isilon," Apr. 23, 2007, 1 page (Exhibit C).
- Taneja Group Technology Analysts, "The Power of InfiniBand and Clustered Storage", Dec. 2005, 8 pages (Exhibit E).
- Arthur S. Rose, Letter to Steven M. Bauer, Aug. 31, 2007, 4 pages.
- Ramez Elmasri, Fundamentals of Database Systems 3rd ed, Sep. 30, 1999, Addison-Wesley, pp. 155, 169, 171, 172, 173, 178, 181, 182, 381, 771.
- MCSA/MSCE/MCDBA Self Paced Training Kit: MS SQL Server 2000 System Administration, May 14, 2003, Microsoft Press, Second Edition, 12 pages.
- Supp EPO Search Report, App. No. EP 03 78 3599; dated Apr. 29, 2008, 2 pages.
- Levy E., Ed.—Institute of Electrical and Electronics Engineers: "Incremental restart (database management)" Proc. Int. Conf. Data Eng., Kobe, JP, Apr. 8-12, 1991; IEEE Comp. Soc. Press, U.S., vol. Conf. 7, Apr. 8, 1991, pp. 640-648.
- Haerder T et al: "Principles of Transaction-Oriented Database Recovery" ACM Computing Surveys, ACM, New York, NY, US, vol. 15, No. 4, Dec. 1, 1983, pp. 287-317.
- Gray J et al: "Transaction processing: concepts and techniques (excerpt), Passage" Transaction Processing: Concepts and Techniques, Jan. 1, 1993, pp. 373-445.
- Garcia-Molina H et al: "Database System Implementation, passage" Database System Implementation, Jan. 1, 2000, pp. I-V, 423-509.
- Wedde H. F., et al.: "A universal framework for managing metadata in the distributed Dragon Slayer System", Proceedings of the Euromicro Conference. Euromicro. Informatics: Inventing the Future, Proceedings of Euromicro Workshop on Multimedia and Telecommunications, vol. 2, Sep. 5, 2000, pp. 96-101.
- Tanenbaum, Andrew S., Minix Operating System, Japan, Apr. 21, 1989, 1st Edition, pp. 328, 505; Also attached is what Applicant believes is the English language version: Tanenbaum, Andrew S., Operating Systems: Design and Implementation, Prentice-Hall, 1987, pp. 279, 347, 348.
- Yamashita, Hirofumi, et al., "Development of RAID Filesystem VAFS/HR," The Special Interest Group Notes of IPSJ, Japan, Information Processing Society of Japan, Sep. 16, 1994, vol. 94, No. 80, p. 9-16.
- Oct. 8, 2008 European Supplemental Search Report EP 02 75 6944, 6 pages.
- Jan. 23, 2009 Examination Report EP 02 75 6944, 5 pages.
- May 18, 2009 Int'l Search report PCT/US08/059798, 18 pages.
- Oct. 22, 2009 Int'l Preliminary Report on Patentability PCT/US08/059798, 10 pages.
- Keidar, et al.: Increasing the Resilience of Distributed and Replicated Database Systems (1998) pp. 1-25.
- Lougher P. et al., "Scalable storage servers for digital audio and video," International Conference on Storage and Recording Systems, 1994, Keele, UK, Jan. 1, 1994, pp. 140-143.
- Reddy, P. Krishna et al., Reducing the blocking in two phase commit with backup sites Apr. 15, 2003, Elsevier, pp. 39-47.
- IOS Commands, Feb. 6th, 2005, Pantz.org, [online], <web.archive.org/web/20050206184604/http://pantz.org/os/ios/ioscommands.shtml>, pp. 1-12 as printed.
- IP Forwarding with Cisco Routers Lab Exercises, Jun. 15th, 1999 [online], <ws.edu.isoc.org/data1999/1481713642400af6a2de65a/ip_fwd_cisco_config.ppt>, pp. 1-11 as printed.
- Hisayuki Aoki, Parallel Filesystem MFS, IPSJ SIG Technical Reports, Japan, Information Processing Society of Japan, Aug. 26, 1996, vol. 96, No. 79, pp. 31 to 36 (stamped Citation 1 / English Translation attached) total 23 pages.
- Yoshitake Shinkai, Cluster File System: HAMFS, Fujitsu, Japan, Fujitsu Limited, Jul. 9, 1999, vol. 50, No. 4, pp. 210 to 214 (stamped Citation 2 / English Translation attached) total 16 pages.
- Jul. 28, 2008 Non-Final Rejection in U.S. Appl. No. 11/256,310, filed Jan. 30, 2006.
- Nov. 14, 2008 Non-Final Rejection in U.S. Appl. No. 11/256,310, filed Jan. 30, 2006.
- Dec. 16, 2008 Response to Nov. 14, 2008 Non-Final Rejection in U.S. Appl. No. 11/256,310, filed Jan. 30, 2006.
- Mar. 23, 2009 Notice of Allowance in U.S. Appl. No. 11/256,310, filed Jan. 30, 2006.
- Jim Gray and Andreas Reuter, Transaction Processing: Concepts and Techniques, Morgan Kaufmann, 1993, pp. 1-328.
- Jim Gray and Andreas Reuter, Transaction Processing: Concepts and Techniques, Morgan Kaufmann, 1993, pp. 329-664.
- Jim Gray and Andreas Reuter, Transaction Processing: Concepts and Techniques, Morgan Kaufmann, 1993, pp. 665-1105.
- Sape Mullender, editor, Distributed Systems (2nd Ed.), ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1993, pp. 1-327.
- Sape Mullender, editor, Distributed Systems (2nd Ed.), ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1993, pp. 328-604.
- Gerhard Weikum, et al., Transactional Information Systems: Theory, Algorithms, and the Practice of Concurrency Control and Recovery, Morgan Kaufmann, 2002, pp. 1-450.
- Gerhard Weikum, et al., Transactional Information Systems: Theory, Algorithms, and the Practice of Concurrency Control and Recovery, Morgan Kaufmann, 2002, pp. 451-863.
- Lougher P. et al., "Scalable storage servers for digital audio and video," International Conference on Storage and Recording Systems, 1994, Keele, UK, Jan. 1, 1994, pp. 140-143.
- Reddy, P. Krishna et al., Reducing the blocking in two phase commit with backup sites Apr. 15, 2003, Elsevier, pp. 39-47.
- IOS Commands, Feb. 6th, 2005, Pantz.org, [online], <web.archive.org/web/20050206184604/http://pantz.org/os/ios/ioscommands.shtml>, 12 pages.
- IP Forwarding with Cisco Routers Lab Exercises, Jun. 15th, 1999 [online], <ws.edu.isoc.org/data1999/1481713642400af6a2de65a/ip_fwd_cisco_config.ppt>, 11 pages.
- Hisayuki Aoki, Parallel Filesystem MFS, IPSJ SIG Technical Reports, Japan, Information Processing Society of Japan, Aug. 26, 1996, vol. 96, No. 79, pp. 31 to 36 (stamped Citation 1 / English Translation attached, 23 pages).
- Yoshitake Shinkai, Cluster File System: HAMFS, Fujitsu, Japan, Fujitsu Limited, Jul. 9, 1999, vol. 50, No. 4, pp. 210 to 214 (stamped Citation 2 / English Translation attached, 16 pages).
- Duchamp, Dan: "Analysis of Transaction Management Performance", ACM Symposium on Operating Systems Principles, Dec. 1989, pp. 177-190.
- Silaghi, Raul et al.: "Porting OMTTs to CORBA", Lecture Notes in Computer Science, vol. 2888/2003, Nov. 2003, pp. 1521-1542.
- Dorai et al.: "Transport Threads: Resource Sharing in SMT Processor for High Single-Thread Performance", proceedings of the 2002 International Conference on Parallel Architectures and Complication Techniques, Sep. 2002, 12 pages.
- Peterson, Zachary Nathaniel Joseph, "Data Placement for Copy-On-Write Using Virtual Contiguity," Master of Science in Computer Science Thesis, University of California, Santa Cruz, Sep. 2002, 67 pages.

* cited by examiner

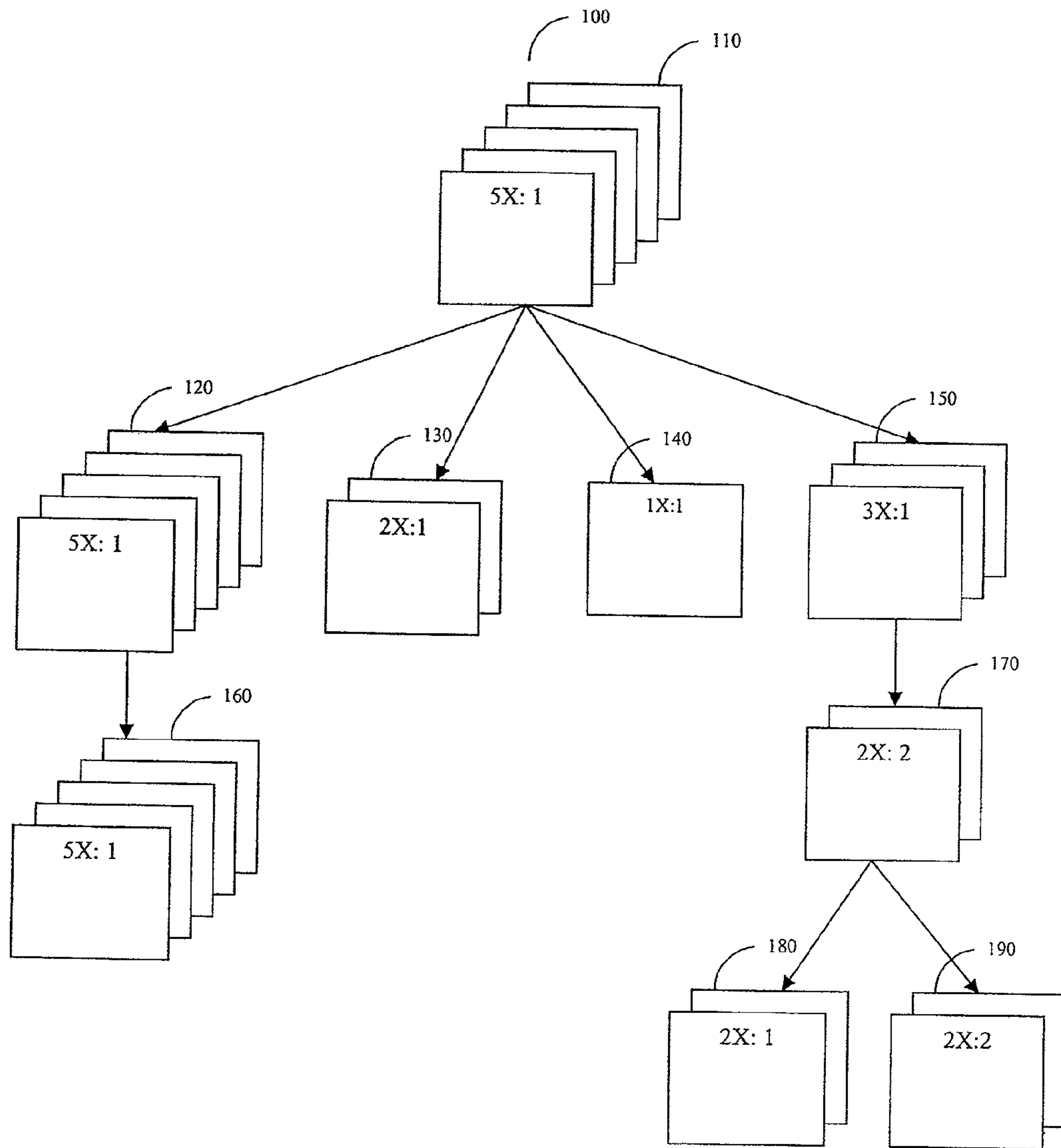


FIGURE 1A

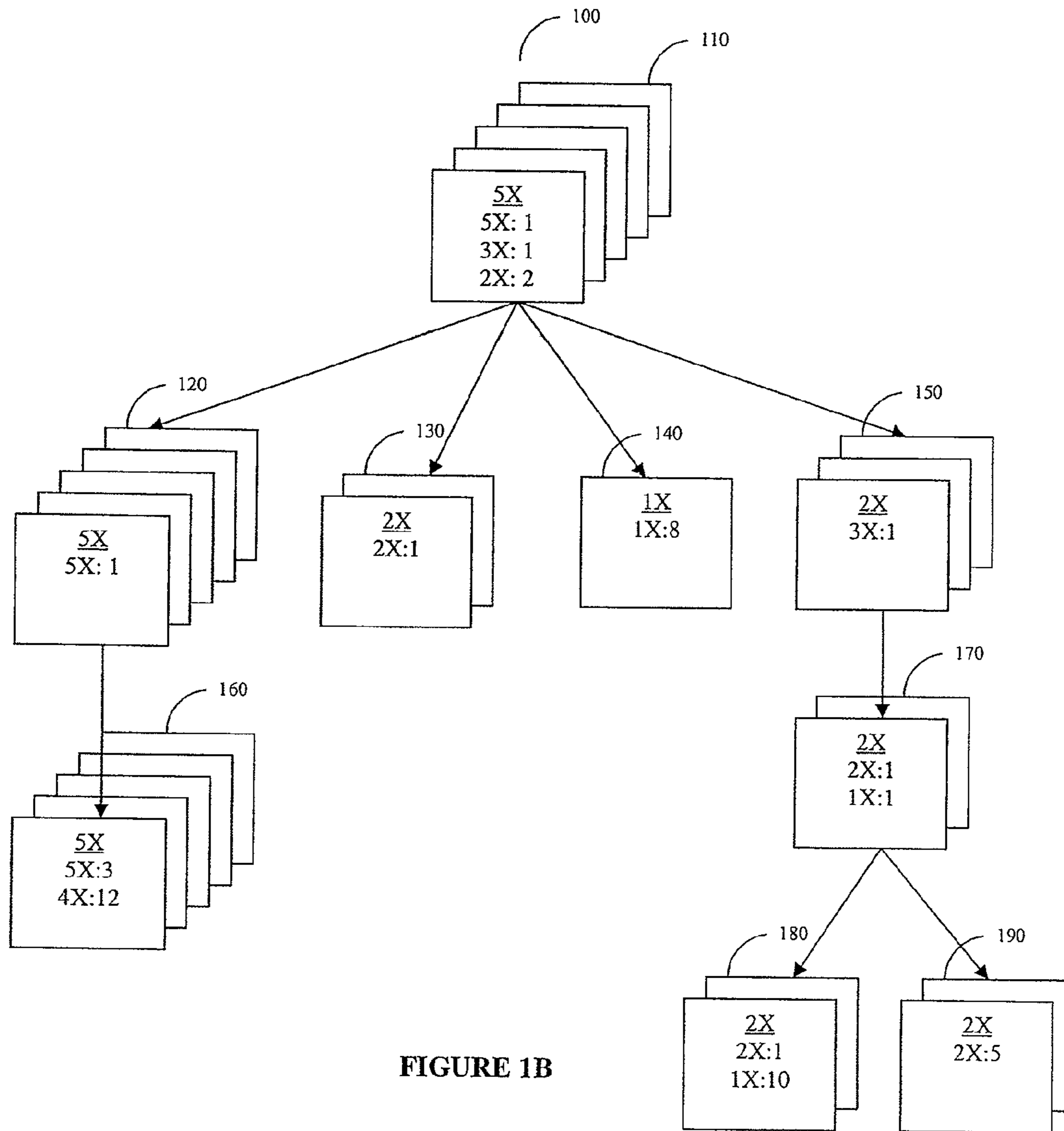


FIGURE 1B

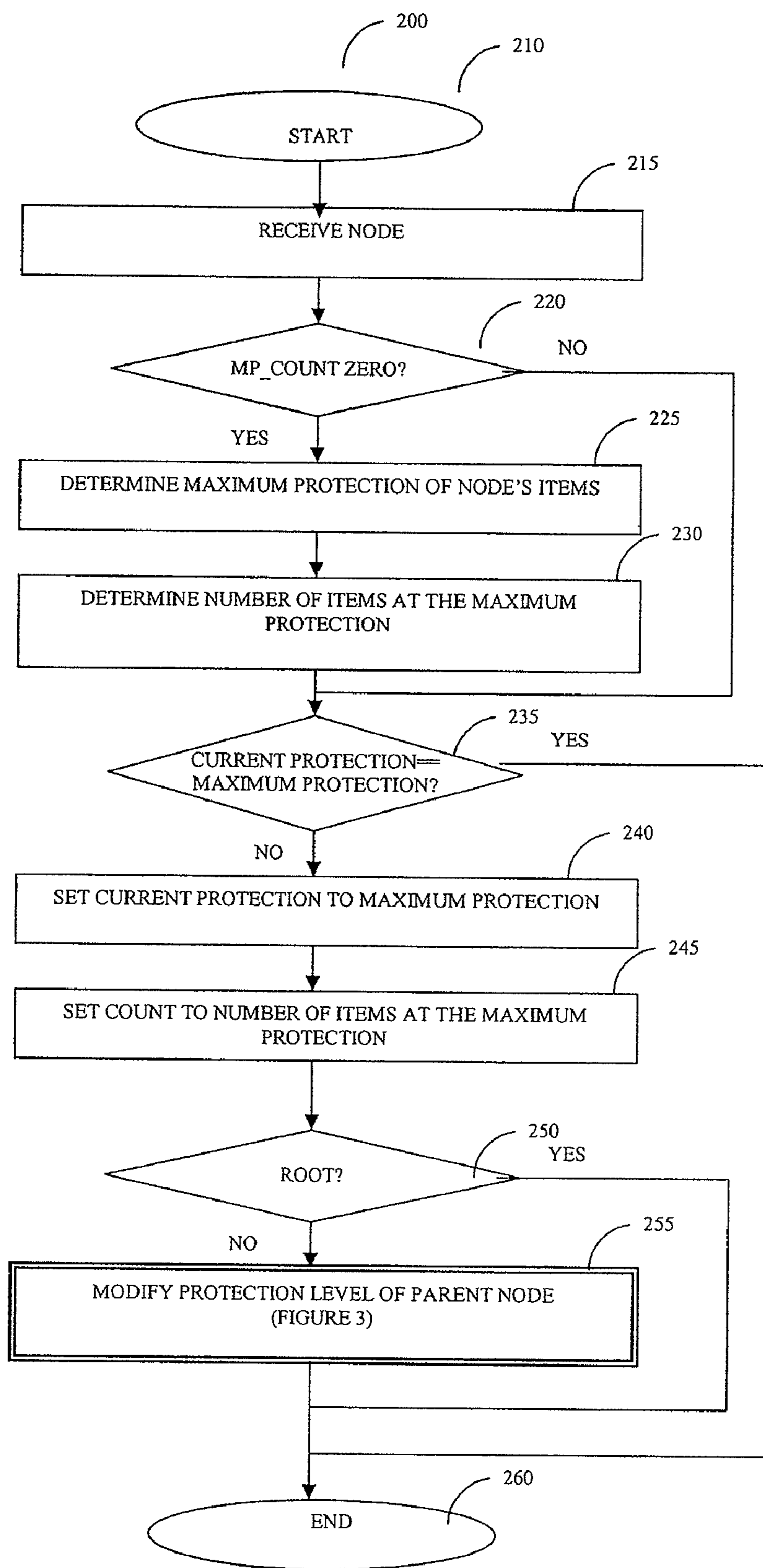


FIGURE 2

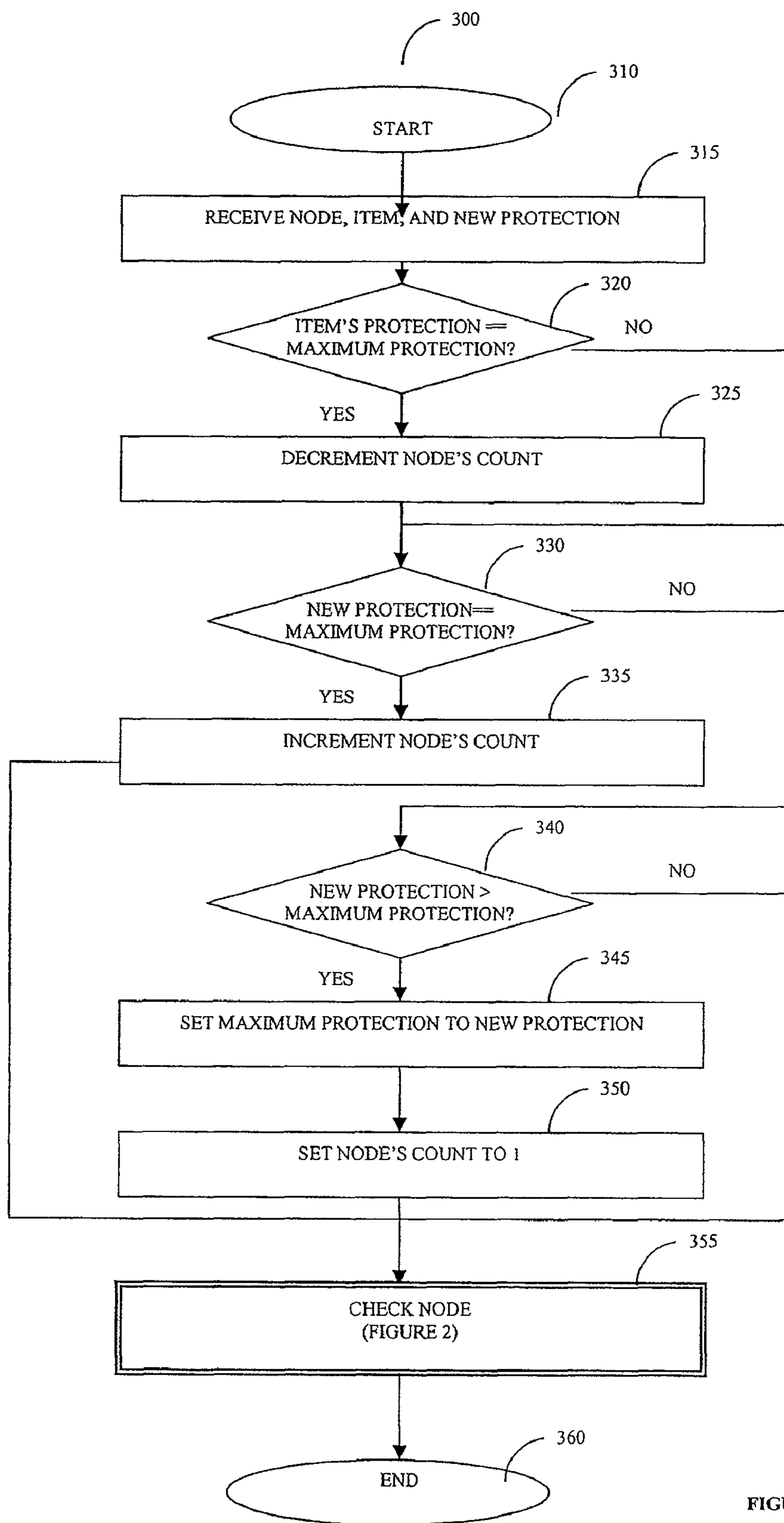


FIGURE 3

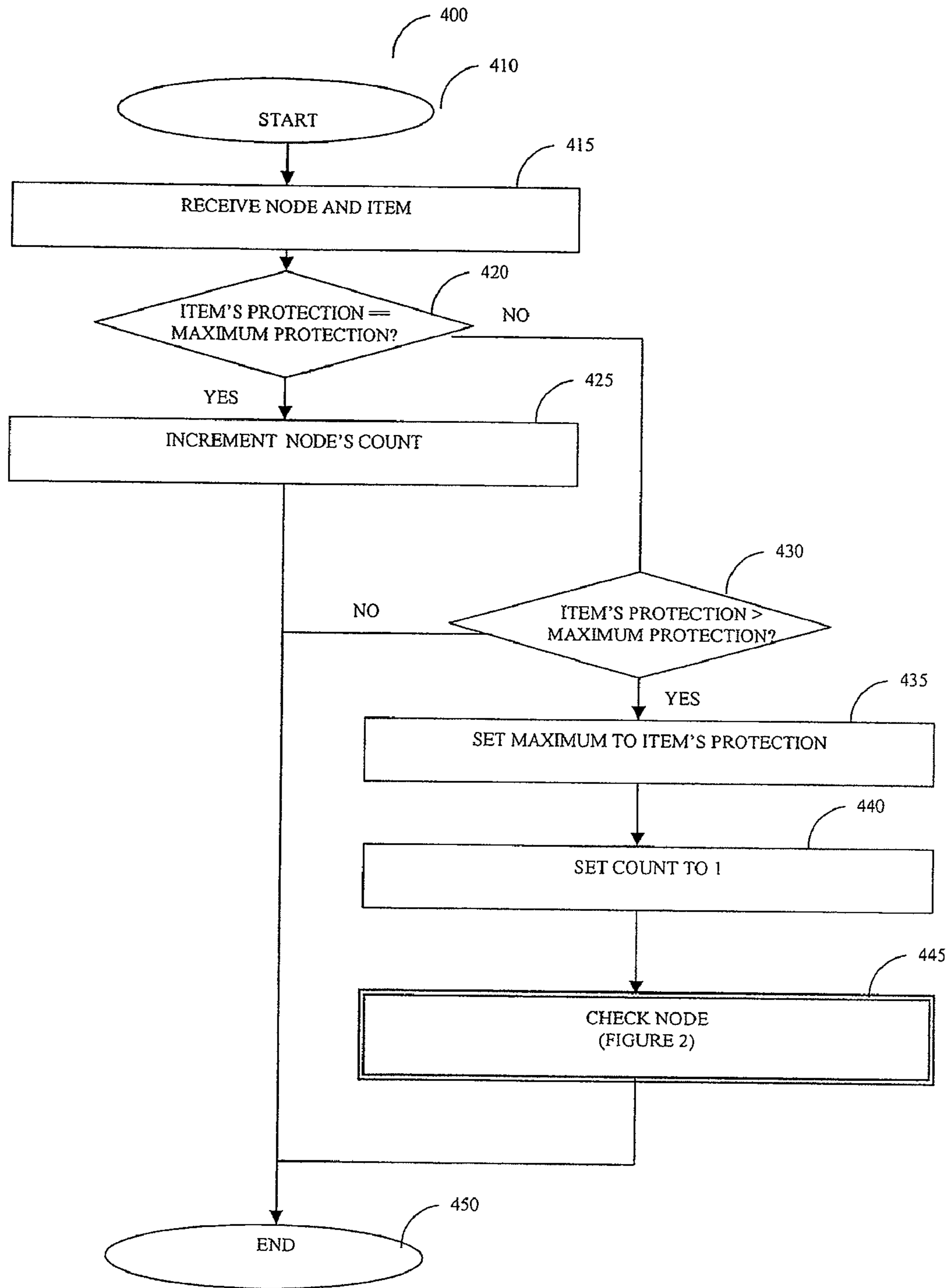


FIGURE 4

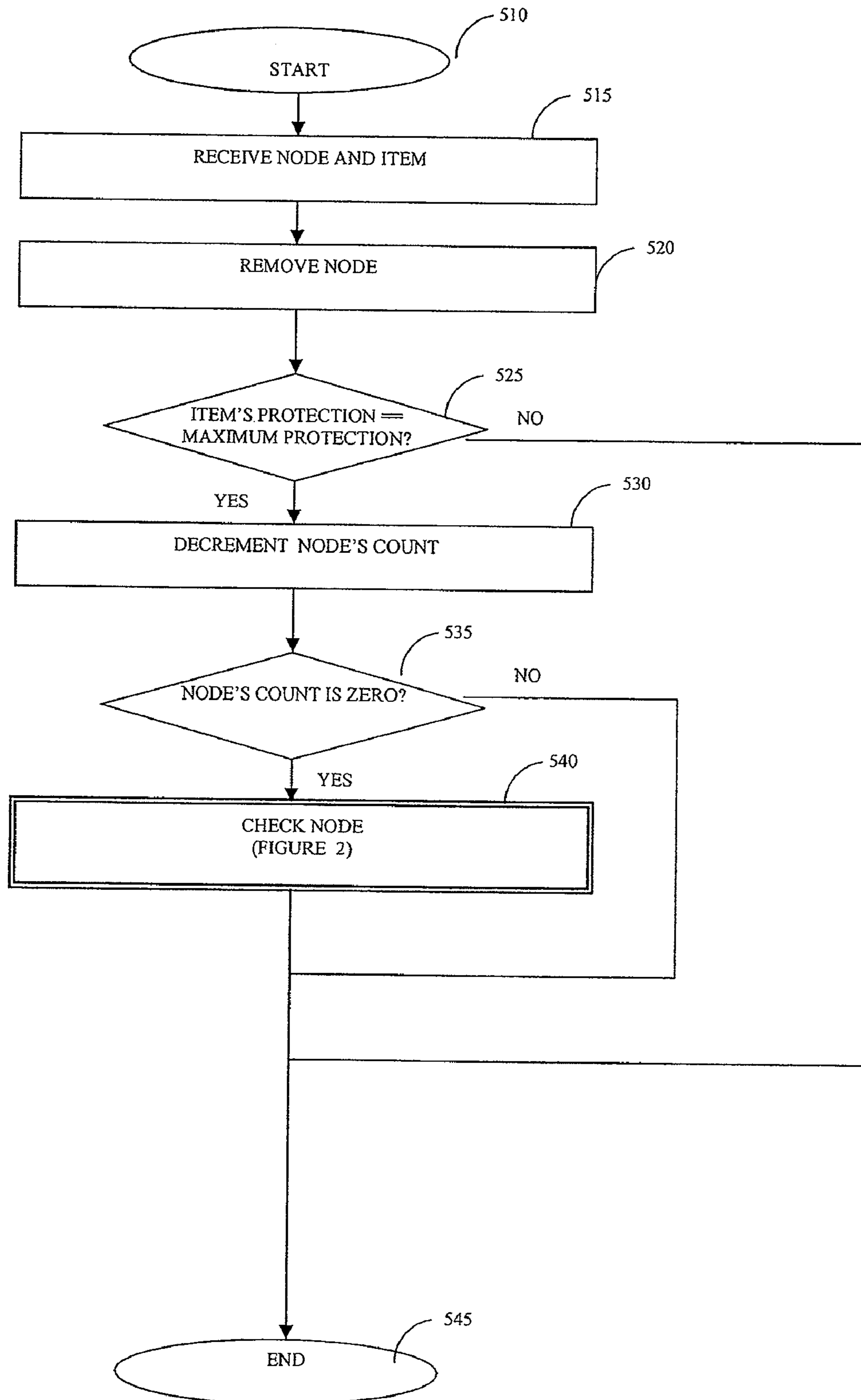


FIGURE 5

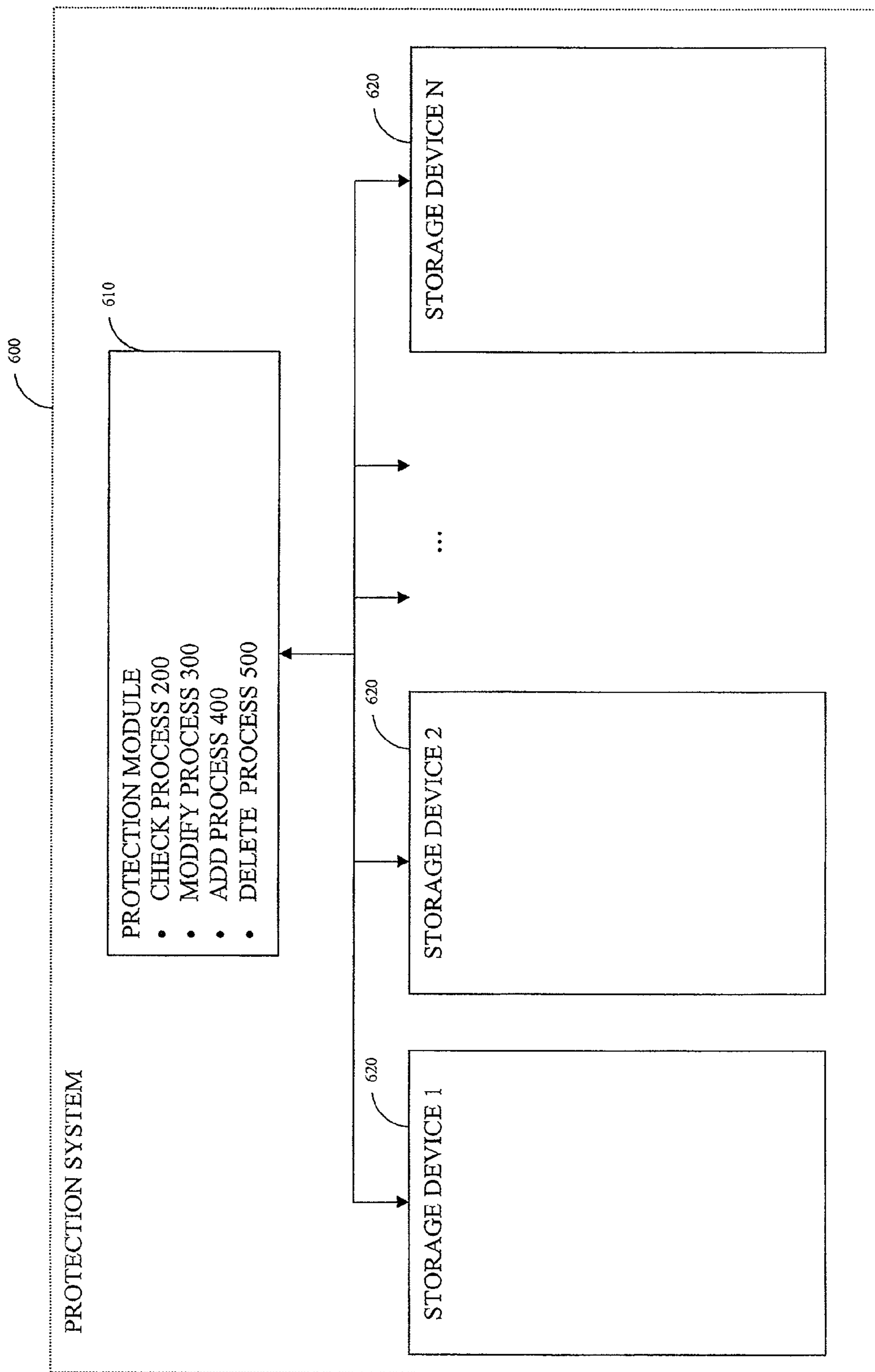


Figure 6

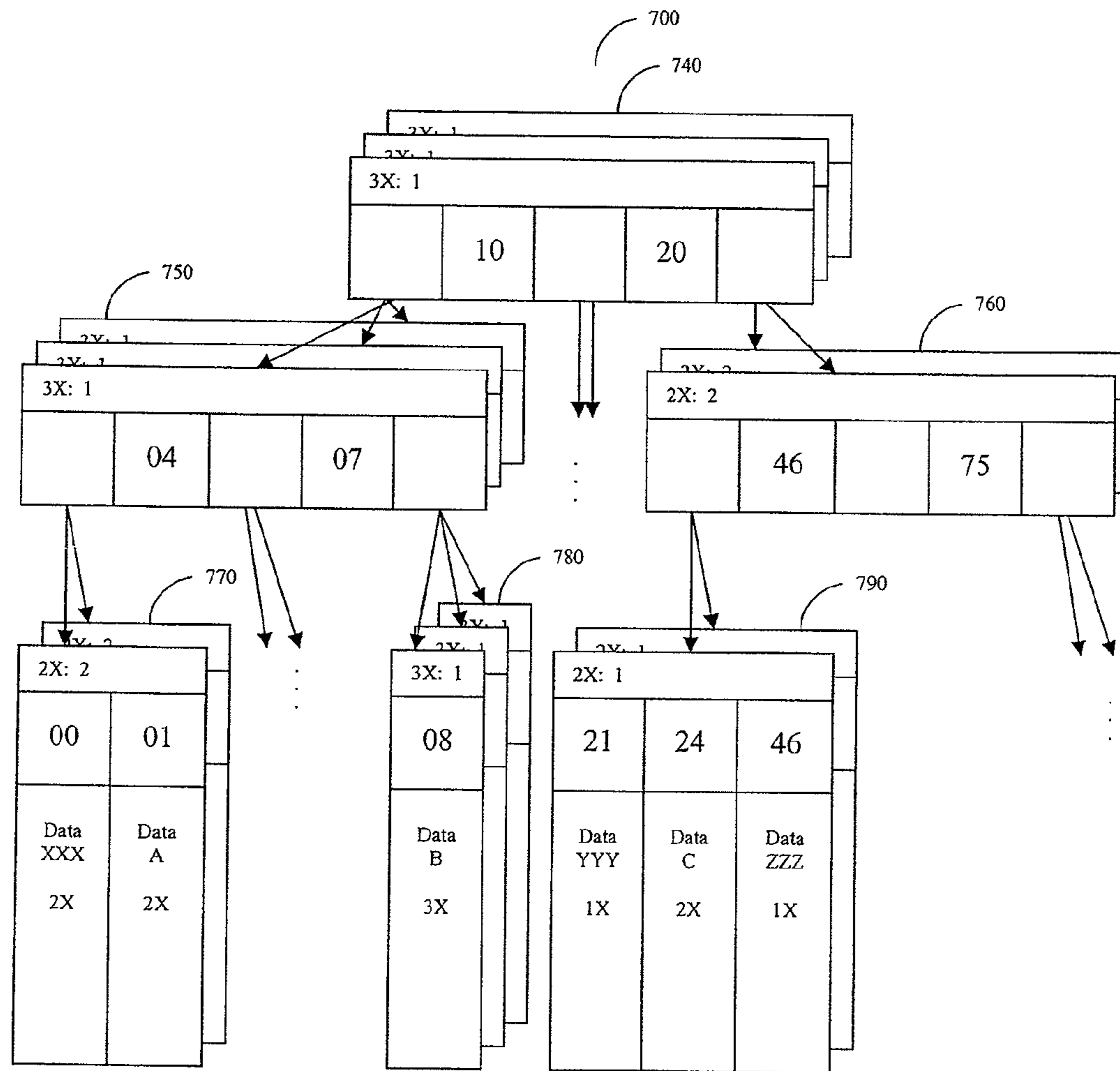


FIGURE 7

SYSTEMS AND METHODS FOR PROVIDING VARIABLE PROTECTION

REFERENCE TO RELATED APPLICATIONS

This application is a continuation of and claims benefit of priority under 35 U.S.C. §120 to U.S. patent application Ser. No. 11/256,410 now U.S. Pat. No. 7,551,572, filed Oct. 21, 2005, entitled "SYSTEMS AND METHODS FOR PROVIDING VARIABLE PROTECTION," which is hereby incorporated by reference herein in its entirety. The present disclosure relates to U.S. patent application Ser. No. 11/255,817, titled "SYSTEMS AND METHODS FOR DISTRIBUTED SYSTEM SCANNING," U.S. patent application Ser. No. 11/255,346, titled "SYSTEMS AND METHODS FOR MANAGING CONCURRENT ACCESS REQUESTS TO A SHARED RESOURCE," U.S. patent application Ser. No. 11/255,818, titled "SYSTEMS AND METHODS FOR MAINTAINING DISTRIBUTED DATA," U.S. patent application Ser. No. 11/256,317, titled "SYSTEMS AND METHODS FOR USING EXCITEMENT VALUES TO PREDICT FUTURE ACCESS TO RESOURCES," and U.S. patent application Ser. No. 11/255,337 titled "SYSTEMS AND METHODS FOR ACCESSING AND UPDATING DISTRIBUTED DATA," each filed on Oct. 21, 2005 and each hereby incorporated by reference herein in their entirety.

FIELD OF THE INVENTION

This invention relates to systems and methods for providing protection within a data structure where the objects of the data structure have different protection requirements.

BACKGROUND

Various protection techniques may be used to protect data. These techniques may include, for example, error correction, redundancy, and mirroring so that if one set of the data is not inaccessible, the data can be recovered. It is often difficult to determine what level of protection is required for the data.

SUMMARY

Systems and methods are disclosed that determine what level of protection is required for a data structure where the objects of the data structure have different protection requirements. In one embodiment, the data structure is protected by protecting objects of the data structure at the same or greater protection level as their children. In another embodiment, the objects store data to track the protection requirements of their children.

In one embodiment, a method is provided that protects data of variable protection levels. The method may include, for each leaf node, determining a protection level for the leaf node; for each non-leaf node, determining a protection level for each non-leaf node based on a maximum of the protection levels of each child of each of the non-leaf nodes; protecting each of the leaf nodes at the protection level determined for each of the leaf nodes; and protecting each of the non-leaf nodes at the protection level determined for each of the non-leaf nodes.

In another embodiment, a protection system is provided for protecting data of variable protection levels. The system may include a set protection level module configured to determine protection levels for each of a set of leaf nodes and determine protection levels for each of a set of non-leaf nodes based on a maximum of the protection levels of children of each of the

non-leaf nodes; and an implement protection level module, in communication with the set protection level module, configured to protect each leaf node at the determined protection level and protect each non-leaf node at the determined protection level.

In another embodiment, a method is provided that updates data of variable protection levels. The method may include determining a maximum protection level of an updated leaf node, wherein the updated leaf node comprises a plurality of data sets; protecting the updated leaf node at the maximum protection level; determining a count of data sets that are at the maximum protection level; and for each ascendant of the updated leaf node, determining a maximum protection level of each ascendant, determining a count of children of each ascendant that are at the maximum protection level, and protecting each ascendant at the maximum protection level.

In another embodiment, a system is provided for updating data stored using variable protection levels. The system may include an update leaf module configured to determine a maximum protection level of an updated leaf node, wherein the updated leaf node comprises a plurality of data sets, and protect the updated leaf node at the maximum protection level; an update non-leaf module in communication with the update leaf module, the update non-leaf module configured to, for each ascendant of the updated leaf node, determine a maximum protection level of each ascendant and protect each ascendant at a protection level greater than or equal to the maximum protection level.

For purposes of summarizing this invention, certain aspects, advantages and novel features of the invention have been described herein. It is to be understood that not necessarily all such advantages may be achieved in accordance with any particular embodiment of the invention. Thus, the invention may be embodied or carried out in a manner that achieves or optimizes one advantage or group of advantages as taught herein without necessarily achieving other advantages as may be taught or suggested herein.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1A illustrates one embodiment of a high-level block diagram of one embodiment of a mirrored tree.

FIG. 1B illustrates another embodiment of a high-level block diagram of one embodiment of a mirrored tree.

FIG. 2 illustrates one embodiment of a flow chart of a check protection information process.

FIG. 3 illustrates one embodiment of a flow chart of an add node process.

FIG. 4 illustrates one embodiment of a flow chart of a delete node process.

FIG. 5 illustrates one embodiment of a flow chart of a modify node process.

FIG. 6 illustrates one embodiment of a protection system.

FIG. 7 illustrates one embodiment of a high-level block diagram of one embodiment of a mirrored index tree.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

Systems and methods which represent one embodiment and example application of the invention will now be described with reference to the drawings. Variations to the systems and methods which represent other embodiments will also be described.

For purposes of illustration, some embodiments will be described in the context of a tree and example environments in which the tree may be used will also be described. The

inventors contemplate that the present invention is not limited by the type of environment in which the systems and methods are used, and that the systems and methods may be used in various environments. It is also recognized that in other embodiments, the systems and methods may be implemented as a single module and/or implemented in conjunction with a variety of other modules and the like. Moreover, the specific implementations described herein are set forth in order to illustrate, and not to limit, the invention. The scope of the invention is defined by the appended claims.

These and other features will now be described with reference to the drawings summarized above. The drawings and the associated descriptions are provided to illustrate embodiments of the invention and not to limit the scope of the invention. Throughout the drawings, reference numbers may be re-used to indicate correspondence between referenced elements. In addition, the first digit of each reference number generally indicates the figure in which the element first appears.

I. Overview

The systems and methods of the present invention provide protection for data structures with nodes of various protection levels. In one embodiment, data is stored in a tree where nodes without children are referred to as leaf nodes and the top node of the tree is referred to as a root node. Nodes of the tree may be protected at various protection levels. In one embodiment, the various protection levels represent the number of mirrored copies of the nodes.

In some embodiments, the nodes have at least the maximum protection level as their children. Accordingly, when a modification is made to one node in a tree, the protection level of that node and its ancestors (e.g., parent node, grandparent node, etc.) are checked to determine whether their protection levels need to be updated. For example, when a node is added to the tree, the protection level for that node is determined, the appropriate number of copies of the node are created, and the node's ancestors are checked to determine whether their protection levels need to be updated. When a node is deleted from the tree, the node's ascendants are checked to determine whether their protection levels need to be updated.

II. Mirrored Tree

A. Example Mirrored Tree

FIG. 1A illustrates an example mirrored tree **100** that includes nine nodes **110**, **120**, **130**, **140**, **150**, **160**, **170**, **180**, **190**. Node **110** is the root node, and nodes **130**, **140**, **160**, **180**, **190** are leaf nodes (because they have no children). Root node **110**, node **120**, and node **160** are mirrored five times (e.g., 5×); node **150** and node **170** are mirrored three times (e.g., 3×); node **130**, node **180**, and node **190** are mirrored two times (e.g., 2×); and node **140** is mirrored one time (e.g., 1×). In one embodiment, mirrored copies of a node are distributed and stored among a set of devices.

In accordance with one embodiment, each node is at a protection level at least the maximum protection level of its children. For example, node **170** has two children, both with protection level 2× **180**. Thus, the maximum protection level of the children of root node **170** is 2×. Accordingly, node **170** is protected at protection level 2× which is at least the maximum protection level of its children. As another example, root node **110** has four children, one with protection level 5× **120**, one with protection level 3× **150**, one with protection level 2× **130**, and one with protection level 1× **140**. Thus, the maximum protection level of the children of root node **110** children is 5×. Accordingly, root node **110** is protected at protection level 5× which is at least the maximum protection level of its children.

B. Protection Level Data

In one embodiment, the different protection levels are managed using protection level information. Protection level information may include, for example, the protection level of a node's children, the maximum protection level of the node's children, the count of children at each protection level, the count of the children with the maximum protection level, protection levels of data entries, the count of data entries at one or more protection levels, and so forth. In FIG. 1A, the protection level information includes the maximum protection level of the node's children followed by the count of the children with the maximum protection level as follows:

[maximum protection level]:[count of children with the maximum protection level]

For example, the protection level information for the root node **170** is: 5×:1 indicating that the maximum protection level of the root node's children is 5× and the number of children at that protection level is 1. As another example, the protection level information for the node **170** is: 2×:2 indicating that the maximum protection level of the root node's children is 2× and the number of children at that protection level is 2. In FIG. 1A, the leaf nodes provide the maximum protection level of their data entries and the number of data entries at that protection level.

A variety of protection level information may be used. For example, FIG. 1B illustrates the same tree **100** wherein different protection level information is stored. In FIG. 1B, the protection level information includes protection level of the node, the protection levels used by the children, and the count of the children with the each of the protection levels.

For example, the protection level information for the root node **170** includes: 5× indicating that the root node **110** is stored at 5×; 5×:1 indicating that one of the root node's children is stored at the protection level 5×; 3×:1 indicating that one of the root node's children is stored at the protection level 3×; 2×:1 indicating that one of the root node's children is stored at the protection level 2×; and 1×:1 indicating that one of the root node's children is stored at the protection level 5×. As another example, the protection level information for the node **170** includes: 2× indicating that node **170** is stored at protection level 2×; 2×:2 indicating that two of the root node's children are stored at protection level 2×. From this information, the total number of children may be determined by summing the counts. Similarly, protection information for the leaf nodes includes the protection levels of the data entries as well as the number of data entries at that protection level. In the example, the leaf nodes may store multiple data entries, though it is recognized that in other embodiments, the leaf nodes may store one data entry or a fixed number of data entries.

C. Nodes

1. Leaf Nodes

The protection level of the leaf nodes may be predetermined, may be set for each leaf node, and/or may be set for a subset of leaf nodes. In addition, a minimum protection level of the leaf nodes may also be set. In one embodiment, where the leaf nodes store data, the protection level may be determined by using the protection level of the data stored in or referenced by the leaf node. For example, in one embodiment, if leaf node **180** includes two sets of data, Data A and Data B, the protection level of leaf node **180** is MAX (Data A, Data B).

In one embodiment, the protection level information for the leaf node indicates the maximum protection level of the data stored in or referenced by the leaf node. In other embodiments, the protection level information for the leaf node indi-

cates the protection level of the leaf node, such as in FIGS. 1A and 1B. In other embodiments, the protection level information for the leaf node is NULL.

2. Non-Leaf Nodes

In order to preserve the protection of the data sets, non-leaf nodes are protected at a level at least that of the maximum protection level of the children of the non-leaf nodes. For example, the protection level of root node 110 is $\text{MAX}(\text{Node } 120, \text{Node } 130, \text{Node } 140, \text{Node } 150) = \text{MAX}(5\times, 2\times, 1\times, 3\times) = 5\times$. The protection level of node 120 is $\text{MAX}(\text{Node } 160) = \text{MAX}(5\times) = 5\times$. The protection level of node 150 is $\text{MAX}(\text{Node } 170) = \text{MAX}(3\times) = 3\times$. The protection level of node 170 is $\text{MAX}(\text{Node } 180, \text{Node } 190) = \text{MAX}(2\times, 2\times) = 2\times$.

In one embodiment, the protection level information for the non-leaf nodes may include the protection level of a node's children, the maximum protection level of the node's children, the count of children at each protection level, the count of the children with the maximum protection level, and so forth. It is recognized that a variety of protection level information could be used. For example, the protection level information could include the maximum protection level without any counts.

It is recognized that in other embodiments, the tree may track other protection information in addition to the maximum protection level and the maximum protection level count. For example, the tree may also track information about the protection levels of each of the node's children in addition to or instead of the maximum protection level and keep a count of the number of children at each protection level. In other embodiments, the tree may also store information about a subset of the protection levels and counts related to those protection levels.

D. Various Embodiments

It is recognized that a variety of embodiments may be used to implement a data structure with variable protection and that the example provided herein are meant to illustrate and not limit the scope of the invention.

1. Data Structures

While the examples discuss the data as being stored in a tree, it is recognized that a variety of data structures known to those of ordinary skill in the art may be used to organize the data including, for example, balanced trees, binary trees, other trees, graphs, linked lists, heaps, stacks, and so forth.

2. Protection Techniques

In one embodiment, data may be protected using a variety of protection schemes, such as, error correction, redundancy, and so forth. The examples discussed below illustrate an embodiment in which the tree is protected using mirroring, though it is recognized that other protection techniques may be used.

While some mirrored trees may be implemented such that the entire tree is always mirrored the same number of times (e.g., all of the nodes are mirrored two times; all of the nodes are mirrored five times, etc.), such an implementation often requires storing extra, unnecessary copies of nodes. Thus, the systems and methods described herein allow different nodes of the tree to be stored using different protection levels while maintaining the protection levels of descendant nodes. In addition, nodes that do not need any protection do not have to use any protection level.

It is recognized that a combination of techniques may be used to implement the tree. For example, the tree may require that every node is mirrored at least two times but that additional mirroring may be implemented to protect nodes that have children that are mirrored more than two times.

III. Operations

Operations for checking the protection information, modifying protection information, adding an item, and removing an item are set forth below. It is recognized that examples below provide various embodiments of the processes and that other embodiments may be used.

A. Check Protection Information of a Node

To check the protection information of a node of a mirrored tree, a node is received and the maximum protection count is determined. If the maximum protection count for the node is zero, then the maximum protection level of the node's items are determined, and the count is updated to reflect the number of items at that protection level. In one embodiment, items for non-leaf nodes are its children nodes, and items for leaf nodes are the data entries in the leaf node.

One example set of pseudocode to check, and if necessary, update the protection information of a node is as follows:

```

fixup_node(node) {
  if (node.mp_count == 0) {
    node.max_protection = MAX(items' protections)
    node.mp_count = count(items at max protection)
  }
  if (node.current_protection != node.max_protection) {
    set_mirror_count(node, node.max_protection)
    node.current_protection = node.max_protection
    if (node != root)
      modify_item_protection(get_parent(node), &node,
        node.max_protection);
  }
}

```

FIG. 2 illustrates one embodiment of a check process 200. Beginning at a start state 210, the check process 200 proceeds to the next block 215. In block 215, the check process 200 receives a node. It is recognized that in some embodiment the check process 200 may receive a reference to the node, information from the node, a pointer to the node, and so forth. Proceeding to block 220, the check process 200 determines whether the node's count of maximum protection items is zero. If not, then the check process 200 proceeds to block 235. If so, then the check process 200 determines the maximum protection of the node's items 225 and the number of items at that maximum protection 230.

Proceeding to the next block 235, the check process 200 determines whether the current protection of the node is the same as the maximum protection. If so, then the check process 200 proceeds to an end state 260. If not, then the check process 200 sets the current protection of the node to the maximum protection 240 and sets the current maximum count to the count of the number of items at that maximum protection 245, and proceeds to block 250.

In block 250, the check process 200 determines whether the current node is the root. If so, the check process 200 proceeds to the end state 260. If not, then the check process 200 calls the modify process to modify the protection information of the node's parent using the node and the new maximum protection and proceeds to the end state 260.

B. Modifying Protection Information of an Item

To modify protection information of an item, a node, an item is received, and a new protection is received. If the item's previous protection is the same as the node's protection, then the node's maximum protection level count is decremented. If the new protection is the same as the node's protection, then the node's maximum protection level count is incremented. If the new protection is higher than the node's maximum protection, then the node's protection is set to the new protection,

7

and the node's maximum protection count is set to 1. Then, the check protection process is called on the node.

One example set of pseudocode to modify protection information is as follows:

```

modify_item_protection(node, item, new_prot) {
  if (item.protection == node.max_protection)
    --node.mp_count
  if (new_prot == node.max_protection) {
    ++node.mp_count
  } else if (new_prot > node.max_protection) {
    node.max_protection = new_prot
    node.mp_count = 1
  }
  fixup_node(node)
}

```

FIG. 3 illustrates one embodiment of a modify process 300. Beginning at a start state 310, the modify process 300 proceeds to the next block 315. In block 315, the modify process 300 receives a node, an item, and a new protection. It is recognized that in some embodiments, the modify process 300 may receive a reference to the node, information from the node, a pointer to the node, and so forth. Proceeding to block 320, the modify process 300 determines whether the node's maximum protection is equal to the item's protection. If not, then the modify process 300 proceeds to block 330. If so, then the modify process 300 decrements the node's count 325 and proceeds to block 330.

In block 330, the modify process 300 determines whether the new protection is the same as the node's maximum protection. If not so, then the modify process 300 proceeds to block 340. If so, then the modify process 300 increments the node's count 335 and proceeds to block 355.

In block 340, the modify process 300 determines whether the new protection is greater than the node's maximum protection. If not, then the modify process 300 proceeds to block 355. If so, then the modify process sets the node's maximum protection to the new protection 345, sets the node's count to 1, and proceeds to block 355.

In block 355, the modify process 300 calls the check process 200 with the node and proceeds to an end state 360.

C. Adding an Item

To add an item, a node and an item are received. The item is added to the node. For leaf nodes, this would include adding a data entry. For non-leaf nodes, this would include adding a child node. If the item's protection is the same as the node's protection, then the node's count is incremented. If the item's protection is greater than the node's protection, then the node's protection is set to the item's protection, the node's count is set to one, and protection information of the node is checked using, for example, the process described above.

One example set of pseudocode to add an item to the mirrored tree is as follows:

```

add_item(node, item) {
  add_item_to_node(node, item)
  if (item.protection == node.max_protection) {
    ++node.mp_count
  } else if (item.protection > node.max_protection) {
    node.max_protection = item.protection
    node.mp_count = 1
    fixup_node(node)
  }
}

```

8

FIG. 4 illustrates one embodiment of an add process 400. Beginning at a start state 410, the add process 400 proceeds to the next block 415. In block 415, the add process 400 receives a node and an item. It is recognized that in some embodiment the add process 400 may receive a reference to the node and/or item, information from the node and/or item, a pointer to the node and/or item, and so forth. Proceeding to block 420, the add process 400 determines whether the item's protection is equal to the node's maximum protection. If not, then the add process 400 proceeds to block 430. If so, then the add process 400 increments the node's count 425 and proceeds to an end state 450.

In block 430, the add process 400 determines whether the item's protection is greater than the node's maximum protection. If not, then the add process 400 proceeds to the end state 450. If so, then the add process 400 sets the node's maximum protection to the item's protection 435, sets the node's count to 1 440, calls the check process 200 with the node 445, and proceeds to the end state 450.

D. Deleting an Item

To delete an item, a node and an item are received. The item is removed from the node. For leaf nodes this may include removing a data entry. For non-leaf nodes, this may include removing a child node. If the item's protection was the same as the node's protection, then the node's count is decremented. If the node's new count is zero, then the protection information of the node is checked using, for example, the process described above.

One example set of pseudocode to delete an item from the mirrored tree is as follows:

```

remove_item(node, item) {
  remove_item_from_node(node, item)
  if (item.protection == node.max_protection) {
    --node.mp_count
    if (node.mp_count == 0)
      fixup_node(node)
  }
}

```

FIG. 5 illustrates one embodiment of a delete process 500. Beginning at a start state 510, the add process 500 proceeds to the next block 515. In block 515, the delete process 500 receives a node and an item. It is recognized that in some embodiment the add process 500 may receive a reference to the node and/or item, information from the node and/or item, a pointer to the node and/or item, and so forth. Proceeding to block 520, the delete process 500 removes the item from the node. Proceeding to block 525, the delete process determines whether the item's protection is equal to the node's maximum protection. If not, then the delete process 500 proceeds to an end state 540. If so, then the delete process 500 decrements the node's count 530, and proceeds to block 535. In block 535, the delete process 500 determines whether the node's count is zero. If not, then the delete process 500 proceeds to the end state. If so, then the delete process 500 calls the check process 200 with the node 540, and proceeds to the end state 540.

IV. Protection System

FIG. 6 illustrates one embodiment of a protection system 600 which includes a protection module 610 in communication with a set of storage devices 620. In one embodiment, the protection module 610 includes the check process 200, the modify process 300, the add process 400, and the delete process 500.

It is recognized that the module may be located apart from the set of storage devices **620** and/or may be located on one or more of the storage devices **620**. In other embodiments, one or more of these modules may be spread among the set of storage devices **620**.

The protection module **610** communicates with the set of storage devices **620** using a variety of communication techniques that are well known in the art. Such communication may include local communication, remote communication, wireless communication, wired communication, or a combination thereof.

In one embodiment, the set of storage devices **620** may include any type of memory that allows for the storing of nodes, protection level information, and/or count information. In some embodiments, the storage device **620** retains data even after power is turned off, however, in other embodiments, the storage device **620** may be implemented using volatile memory. The storage devices **620** may be, for example, a standard hard drive, a cache, ROM, RAM, flash memory, as well as other types of memory as is known by those of ordinary skill in the art.

In some embodiments, the protection system **600** may be accessible by one or more other systems, modules, and/or users via various types of communication. Such communication may include, for example, the Internet, a private network for a hospital, a broadcast network for a government agency, an internal network of a corporate enterprise, an intranet, a local area network, a wide area network, and so forth. It is recognized that the protection system **600** may be used in a variety of environments in which data is stored. For example, the protection system **600** may be used to store records in a database, content data, metadata, user account data, and so forth.

In one embodiment, the protection system **600** runs on a variety of computer systems such as, for example, a computer, a server, a smart storage unit, and so forth. In one embodiment, the computer may be a general purpose computer using one or more microprocessors, such as, for example, a Pentium processor, a Pentium II processor, a Pentium Pro processor, a Pentium IV processor, an x86 processor, an 8051 processor, a MIPS processor, a Power PC processor, a SPARC processor, an Alpha processor, and so forth. The computer may run a variety of operating systems that perform standard operating system functions such as opening, reading, writing, and closing a file. It is recognized that other operating systems may be used, such as, for example, Microsoft® Windows® 3.X, Microsoft® Windows 98, Microsoft® Windows® 2000, Microsoft® Windows® NT, Microsoft® Windows® CE, Microsoft® Windows® ME, Palm Pilot OS, Apple® MacOS®, Disk Operating System (DOS), UNIX, IRIX, Solaris, SunOS, FreeBSD, Linux®, or IBM® OS/2® operating systems.

As used herein, the word module refers to logic embodied in hardware or firmware, or to a collection of software instructions, possibly having entry and exit points, written in a programming language, such as, for example, C or C++. A software module may be compiled and linked into an executable program, installed in a dynamic link library, or may be written in an interpreted programming language such as BASIC, Perl, or Python. It will be appreciated that software modules may be callable from other modules or from themselves, and/or may be invoked in response to detected events or interrupts. Software instructions may be embedded in firmware, such as an EPROM. It will be further appreciated that hardware modules may be comprised of connected logic units, such as gates and flip-flops, and/or may be comprised of programmable units, such as programmable gate arrays or

processors. The modules described herein are preferably implemented as software modules, but may be represented in hardware or firmware.

It is also recognized that in some embodiments, the systems and methods may be implemented as a single module and/or implemented in conjunction with a variety of other modules and the like. Moreover, the specific implementations described herein are set forth to illustrate, and not to limit, the present disclosure.

V. Example Environment

The following provides an example environment in which a distributed mirrored index tree may be used. It is recognized that the systems and methods disclosed herein are not limited to the example environment and that the example is only meant to illustrate embodiments of the invention.

A. Indexing Tree

In one embodiment, the indexing tree is an index tree wherein the copies of the nodes of the tree are stored across devices in a distributed system. The nodes of the indexing tree are stored using at least the same level of protection of data stored in the indexing data structure. In one embodiment, nodes that are stored on an offline device are restored, offline devices that come back online are merged into the distributed system and given access to the index tree, and the index tree is traversed to locate and restore nodes that are stored on offline devices. Additional description of an indexing tree structure is set forth in U.S. Patent Applications entitled "Systems and Methods for Maintaining Distributed Data," and "Systems and Methods for Updating Distributed Data," filed concurrently herewith, which are incorporated by reference above.

In one embodiment, data stored in the tree includes metadata that represents the protection level of the data. In addition, each node, including the root node, the internal nodes, and the leaf nodes, also includes protection information. FIG. 7 illustrates one embodiment of an index, where the index tree includes different mirroring levels, such that different nodes in the index tree are mirrored a different number of times. For example, Data B is stored with a protection level of 3x. Accordingly, the branches of the index tree **740**, **750** that lead to Data B are also protected at a protection level of at least 3x.

1. Leaf Nodes

In the example, leaf node **770** has 2 copies, leaf node **780** has 3 copies, and leaf node **790** has 2 copies. In addition, the number of data entries that have the maximum level of protection for leaf node **770** is 2; the number of data entries that have the maximum level of protection for leaf node **780** is 1; and the number of data entries that have the maximum level of protection for leaf node **790** is 1.

2. Non-Leaf Nodes

In the example, the protection level of internal node **750** is $\text{MAX}(\text{Leaf Node } 770, \text{ Leaf Node } 780, \text{ Other Children Nodes}) = \text{MAX}(2x, 3x, 2x) = 3x$; the protection level of internal node **760** is $\text{MAX}(\text{Leaf Node } 790, \text{ Other Children Nodes}) = \text{MAX}(2x, 2x) = 2x$; and the protection level of root node **740** is $\text{MAX}(\text{Internal Node } 750, \text{ Internal Node } 760, \text{ Other Children Nodes}) = \text{MAX}(3x, 2x, 2x) = 3x$. In addition, the number of children that have the maximum level of protection for internal node **750** is 1; the number of children that have the maximum level of protection for internal node **760** is 2; and the number of children that have the maximum level of protection for root node **740** is 1.

Thus, the index information for each node is protected at least as much as the node's children. In addition, unnecessary replication is avoided. For example, only the nodes in the path

11

to the data set that has a protection level 3× are copied three times, thereby saving on resources while maintaining the integrity of the data.

In one embodiment, the data in the indexing system is used to store identifiers for files and/or directories, and where the data entries provide the actual address of the metadata data structure, or inode, of the files and/or directories.

VI. Conclusion

While certain embodiments of the invention have been described, these embodiments have been presented by way of example only, and are not intended to limit the scope of the present invention. Accordingly, the breadth and scope of the present invention should be defined in accordance with the following claims and their equivalents.

What is claimed is:

1. A method of protecting nodes within a hierarchical data structure at variable protection levels, the hierarchical data structure comprising leaf nodes and non-leaf nodes, the method comprising:

determining a protection level for each of one or more leaf nodes, wherein each protection level corresponds to a level of protection against loss of data, and wherein each level of the protection against the loss of the data corresponds to at least one of a number of mirrored copies, a level of error correction, and a level of parity protection of the leaf node;

determining, by a computer processor, a protection level for a non-leaf node, wherein the non-leaf node has one or more children with corresponding protection levels, and wherein the protection level of the non-leaf node is based on a maximum of the protection levels of current children of the non-leaf node;

identifying an increase in the maximum of the protection levels of the current children of the non-leaf node;

as a result of identifying the increase in the maximum of the protection levels of the current children of the non-leaf node, determining, by a computer processor, an increased protection level for the non-leaf node; and

increasing at least one of a number of mirrored copies of the non-leaf node, a level of error correction of the non-leaf node, and a level of parity protection of the non-leaf node to correspond to the increased protection level for the non-leaf node.

2. The method of claim 1, further comprising:

storing the protection level for each of the one or more leaf nodes; and

storing the protection level for the non-leaf node.

3. The method of claim 1, wherein each of the one or more leaf nodes is associated with one or more sets of data, and determining the protection level for the each of the one or more leaf nodes is based on a maximum of the protection levels of the one or more sets of the data associated with the leaf nodes.

4. The method of claim 1, further comprising, for the non-leaf node, storing protection level information, the protection level information comprising the number of the current children of the non-leaf node having the protection level of the non-leaf node.

5. The method of claim 4, wherein the protection level information further comprises the number of the current children of the non-leaf node at each protection level other than the protection level of the non-leaf node.

12

6. A protection system for protecting data of variable protection levels, the system comprising:

at least one computer processor;

at least one memory;

a hierarchical data structure stored on the at least one memory and accessible by the at least one computer processor, the hierarchical data structure comprising leaf nodes and non-leaf nodes; and

at least one executable software module, the computer processor configured to execute the at least one executable software module, the at least one executable software module configured to:

determine protection levels for each of one or more of the leaf nodes;

determine a protection level for a non-leaf node, wherein the non-leaf node has one or more children with corresponding protection levels, wherein the protection level for the non-leaf node is based on a maximum of the protection levels of current children of the non-leaf node, wherein each protection level corresponds to a level of protection against loss of data, and wherein each level of the protection against the loss of the data corresponds to at least one of a number of mirrored copies, a level of error correction, and a level of parity protection of the leaf node;

identify an increase in the maximum of the protection levels of the current children of the non-leaf node; and increase at least one of a number of mirrored copies of the non-leaf node, a level of error correction of the non-leaf node, and a level of parity protection of the non-leaf node to correspond to the identified increase in the maximum of the protection levels of the current children of the non-leaf node.

7. The system of claim 6, wherein the at least one executable software module is further configured to:

store a protection level indicator on the at least one memory for each of the leaf nodes; and

store a protection level indicator on the at least one memory for the non-leaf node.

8. The system of claim 6, wherein each of the leaf nodes is associated with one or more sets of data, and the at least one executable software module is further configured to determine the protection levels for each of the one or more leaf nodes based on a maximum of the protection levels of the one or more sets of the data associated with the leaf node.

9. The system of claim 6, wherein the at least one executable software module is further configured to, for the non-leaf node, store protection level information, the protection level information comprising the number of the current children of the non-leaf node having the protection level of the non-leaf node.

10. The system of claim 9, wherein the protection level information further comprises the number of the current children of the non-leaf node at each protection level other than the protection level of the non-leaf node.

11. A system for updating data stored using variable protection levels, the system comprising:

at least one computer processor;

at least one memory;

a hierarchical data structure stored on the at least one memory and accessible by the at least one computer processor, the hierarchical data structure comprising leaf nodes and non-leaf nodes, the leaf nodes comprising one or more references to one or more data sets; and

at least one executable software module, the computer processor configured to execute the at least one execut-

13

able software module, the at least one executable software module configured to:

determine a first protection level of a leaf node based on a maximum of protection levels of the data sets referenced by the leaf node, wherein each protection level corresponds to a level of protection against loss of the data, and wherein each level of the protection against the loss of the data corresponds to at least one of a number of mirrored copies, a level of error correction, and a level of parity protection of the leaf node;

determine an increased protection level of an ancestor node of the leaf_node, wherein the ancestor node has one or more children with corresponding protection levels, and wherein the protection level of the ancestor node is based on a maximum of the protection levels of current children of the ancestor node; and

increase at least one of a number of mirrored copies of the ancestor node, a level of error correction of the ancestor

14

node, and a level of parity protection of the ancestor node to correspond to the increased protection level of the ancestor node.

12. The system of claim **11**, wherein the at least one executable software module is further configured to:

store a protection level indicator on the at least one memory for the leaf node; and

store a protection level indicator on the at least one memory for the ancestor node of the leaf node.

13. The system of claim **11**, wherein the at least one executable software module is further configured to, for the ancestor node of the leaf node, store protection level information, the protection level information comprising the number of the current children of the ancestor node having the protection level of the ancestor node.

14. The system of claim **13**, wherein the protection level information further comprises the number of the current children of the ancestor node at each protection level other than the protection level of the ancestor node.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE
CERTIFICATE OF CORRECTION

PATENT NO. : 8,054,765 B2
APPLICATION NO. : 12/484905
DATED : November 8, 2011
INVENTOR(S) : Aaron J. Passey et al.

Page 1 of 1

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

In the Specifications

At Column 6, Line 63, Change “maxim-m” to --maximum--.

In the Claims

At Column 13, Line 13 (Approx.), In Claim 11, change “leaf_node,” to --leaf node,--.

Signed and Sealed this
Twenty-eighth Day of May, 2013



Teresa Stanek Rea
Acting Director of the United States Patent and Trademark Office